I claim:

1. In a computing environment having a connection to a network, computer readable code readable by a computer system in said environment, for enhancing performance of a multithreaded application, comprising:

a plurality of client requests for connections;

a plurality of worker threads;

a subprocess for receiving said plurality of client requests; and

a subprocess for implementing a scheduling heuristic to alleviate over-scheduling of said worker threads.

2. Computer readable code for enhancing performance of a multithreaded application according to Claim 1, wherein:

a first group of said worker threads are active threads, said first group being comprised of changeable ones of said plurality of worker threads, and having a changeable number of said changeable ones, said changeable number being at least one; and

said subprocess for implementing a scheduling heuristic further comprises a subprocess for balancing said changeable number in said first group against a current workload comprised of one or more of said plurality of client requests.

3. Computer readable code for enhancing performance of a multithreaded application according to Claim 2, wherein said subprocess for balancing further comprises using an average delay.

4. Computer readable code for enhancing performance of a multithreaded application according to Claim 3, wherein said subprocess for balancing further comprises using a maximum delay.

5. Computer readable code for enhancing performance of a multithreaded application according to Claim 4, wherein said average delay and said maximum delay are configuration parameters.

6. Computer readable code for enhancing performance of a multithreaded application according to Claim 2, wherein:

a second group of said worker threads are blocked threads, said second group being comprised of ones of said plurality of worker threads which are not in said first group; and

said blocked threads are stored in a Last-In, First-Out queue.

7. In a computing environment having a connection to a network, computer readable code readable by a computer system in said environment, for enhancing performance of a multithreaded application, comprising:

a subprocess for moving connections from a pending connections queue to a first queue when each of said connections are accepted;

a subprocess for moving each of said connections from said first queue to a second queue when an initial data packet arrives for said connection; and a subprocess for assigning a worker thread to each of said connections on said second queue.

8. In a computing environment having a connection to a network, computer readable code readable by a computer system in said environment, for enhancing performance of a multithreaded application, comprising:

a subprocess for receiving input from multiple sources; and

a subprocess for merging said received input onto a single queue for scheduling.

9. Computer readable code for enhancing performance of a multithreaded application according to Claim 8, further comprising:

a subprocess for moving connections from a pending connections queue to a first queue when each of said connections are accepted;

a subprocess for moving each of said connections from said first queue to said single queue when an initial data packet arrives for said connection; and

a subprocess for assigning a worker thread to each of said connections on said single queue.

10. Computer readable code for enhancing performance of a multithreaded application according to Claim 9, wherein said subprocess for scheduling further comprises:

a group of active worker threads comprised of changeable ones of a plurality of worker threads, and having a changeable number of said changeable ones, said changeable number being at least one; and

a subprocess for implementing a scheduling heuristic for balancing said changeable number in said active group against a current workload comprised of said client requests stored on said single queue.

11. In a computing environment having a connection to a network, computer readable code readable by a computer system in said environment, for enhancing performance of a multithreaded application, comprising:

a plurality of persistent connections;

a plurality of worker threads;

a subprocess for binding selected ones of said persistent connections to selected ones of said worker threads, wherein an execution of said subprocess for binding results in a bound connection; and

a subprocess for unbinding selected ones of said bound connections, wherein an execution of said subprocess for unbinding results in an unbound worker thread.

12. Computer readable code for enhancing performance of a multithreaded application according to Claim 11, wherein:

said subprocess for binding further comprises using a 2-stage queue; and

said subprocess for unbinding further comprises using said 2-stage queue.

13. Computer readable code for enhancing performance of a multithreaded application

according to Claim 12, wherein:

said subprocess for binding using said 2-stage queue further comprises:

a subprocess for moving each of said persistent connections to said first stage when an initial data packet arrives for said connection;

a subprocess for moving each of said persistent connections from said second stage to said first stage when data is received for said connection; and

a subprocess for scheduling said persistent connections from said first stage; and

said subprocess for unbinding using said 2-stage queue further comprises:

a subprocess for moving selected ones of said bound connections from said first stage to said second stage when said selected bound connection goes idle;

a subprocess for closing selected ones of said persistent connections in said second stage, responsive to a maximum idle period being exceeded; and

a subprocess for making said unbound worker thread available to said subprocess for binding.

14. Computer readable code for enhancing performance of a multithreaded application according to Claim 13, wherein said subprocess for unbinding further comprises:

a subprocess for closing further selected ones of said persistent connections in said second stage, responsive to exceeding a maximum number of idle connections.

15. A system for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising:

a plurality of client requests for connections;

a plurality of worker threads;

means for receiving said plurality of client requests; and

means for implementing a scheduling heuristic to alleviate over-scheduling of said worker threads.

16. The system for enhancing performance of a multithreaded application according to Claim 15, wherein:

a first group of said worker threads are active threads, said first group being comprised of changeable ones of said plurality of worker threads, and having a changeable number of said changeable ones, said changeable number being at least one; and

said means for implementing a scheduling heuristic further comprises means for balancing said changeable number in said first group against a current workload comprised of one or more of said plurality of client requests.

17. The system for enhancing performance of a multithreaded application according to Claim 16, wherein said means for balancing further comprises using an average delay.

18. The system for enhancing performance of a multithreaded application according to Claim 17, wherein said means for balancing further comprises using a maximum delay.

19. The system for enhancing performance of a multithreaded application according to Claim

18, wherein said average delay and said maximum delay are configuration parameters.

20. The system for enhancing performance of a multithreaded application according to Claim 16, wherein:

a second group of said worker threads are blocked threads, said second group being comprised of ones of said plurality of worker threads which are not in said first group; and

said blocked threads are stored in a Last-In, First-Out queue.

21. A system for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising:

means for moving connections from a pending connections queue to a first queue when each of said connections are accepted;

means for moving each of said connections from said first queue to a second queue when an initial data packet arrives for said connection; and

means for assigning a worker thread to each of said connections on said second queue.

22. A system for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising:

means for receiving input from multiple sources; and

means for merging said received input onto a single queue for scheduling.

23. The system for enhancing performance of a multithreaded application according to Claim

22, further comprising:

means for moving connections from a pending connections queue to a first queue when each of said connections are accepted;

means for moving each of said connections from said first queue to said single queue when an initial data packet arrives for said connection; and

means for assigning a worker thread to each of said connections on said single queue.

24. The system for enhancing performance of a multithreaded application according to Claim 23, wherein said means for scheduling further comprises:

a group of active worker threads comprised of changeable ones of a plurality of worker threads, and having a changeable number of said changeable ones, said changeable number being at least one; and

means for implementing a scheduling heuristic for balancing said changeable number in said active group against a current workload comprised of said client requests stored on said single queue.

25. A system for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising:

a plurality of persistent connections;

a plurality of worker threads;

means for binding selected ones of said persistent connections to selected ones of said worker threads, wherein an execution of said subprocess for binding results in a bound connection; and

means for unbinding selected ones of said bound connections, wherein an execution of said subprocess for unbinding results in an unbound worker thread.

26. The system for enhancing performance of a multithreaded application according to Claim 25, wherein:

said means for binding further comprises using a 2-stage queue; and

said means for unbinding further comprises using said 2-stage queue.

27. The system for enhancing performance of a multithreaded application according to Claim 26, wherein:

said means for binding using said 2-stage queue further comprises:

means for moving each of said persistent connections to said first stage when an initial data packet arrives for said connection;

means for moving each of said persistent connections from said second stage to said first stage when data is received for said connection; and

means for scheduling said persistent connections from said first stage; and

said means for unbinding using said 2-stage queue further comprises:

means for moving selected ones of said bound connections from said first stage to said second stage when said selected bound connection goes idle;

means for closing selected ones of said persistent connections in said second stage, responsive to a maximum idle period being exceeded; and means for making said unbound worker thread available to said subprocess for binding.

28. The system for enhancing performance of a multithreaded application according to Claim 27, wherein said means for unbinding further comprises:

means for closing further selected ones of said persistent connections in said second stage, responsive to exceeding a maximum number of idle connections.

29. A method for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising the steps of:

receiving a plurality of client requests for connections; and

implementing a scheduling heuristic to alleviate over-scheduling of a plurality of worker threads to said plurality of client requests.

30. The method for enhancing performance of a multithreaded application according to Claim 29, wherein:

a first group of said worker threads are active threads, said first group being comprised of changeable ones of said plurality of worker threads, and having a changeable number of said changeable ones, said changeable number being at least one; and

said implementing a scheduling heuristic step further comprises balancing said changeable number in said first group against a current workload comprised of one or more of said plurality of client requests.

31. The method for enhancing performance of a multithreaded application according to Claim 30, wherein said balancing step further comprises using an average delay.

32. The method for enhancing performance of a multithreaded application according to Claim 31, wherein said balancing step further comprises using a maximum delay.

33. The method for enhancing performance of a multithreaded application according to Claim 32, wherein said average delay and said maximum delay are configuration parameters.

34. The method for enhancing performance of a multithreaded application according to Claim 30, wherein:

a second group of said worker threads are blocked threads, said second group being comprised of ones of said plurality of worker threads which are not in said first group; and

said blocked threads are stored in a Last-In, First-Out queue.

35. A method for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising the steps of:

moving connections from a pending connections queue to a first queue when each of said connections are accepted;

moving each of said connections from said first queue to a second queue when an initial data packet arrives for said connection; and assigning a worker thread to each of said connections on said second queue.

36. A method for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising the steps of:

receiving input from multiple sources; and

merging said received input onto a single queue for scheduling.

37. The method for enhancing performance of a multithreaded application according to Claim 36, further comprising the steps of:

moving connections from a pending connections queue to a first queue when each of said connections are accepted;

moving each of said connections from said first queue to said single queue when an initial data packet arrives for said connection; and

assigning a worker thread to each of said connections on said single queue.

38. The method for enhancing performance of a multithreaded application according to Claim 37, further comprising:

a group of active worker threads comprised of changeable ones of a plurality of worker threads, and having a changeable number of said changeable ones, said changeable number being at least one; and

wherein said scheduling step further comprises:

implementing a scheduling heuristic for balancing said changeable number in said active group against a current workload comprised of said client requests stored on said single queue.

39. A method for enhancing performance of a multithreaded application in a computing environment having a connection to a network, comprising the steps of:

binding selected ones of a plurality of persistent connections to selected ones of a plurality of worker threads, wherein an execution of said binding step results in a bound connection; and

unbinding selected ones of said bound connections, wherein an execution of said unbinding step results in an unbound worker thread.

40. The method for enhancing performance of a multithreaded application according to Claim 39, wherein:

said binding step further comprises using a 2-stage queue; and

said unbinding step further comprises using said 2-stage queue.

41. The method for enhancing performance of a multithreaded application according to Claim 40, wherein:

said binding using said 2-stage queue step further comprises the steps of:

moving each of said persistent connections to said first stage when an initial data packet arrives for said connection;

moving each of said persistent connections from said second stage to said first stage when data is received for said connection; and

scheduling said persistent connections from said first stage; and said unbinding using said 2-stage queue step further comprises the steps of:

moving selected ones of said bound connections from said first stage to said second stage when said selected bound connection goes idle;

closing selected ones of said persistent connections in said second stage, responsive to a maximum idle period being exceeded; and

making said unbound worker thread available to said subprocess for binding.

42. The method for enhancing performance of a multithreaded application according to Claim 41, wherein said unbinding step further comprises the step of:

closing further selected ones of said persistent connections in said second stage, responsive to exceeding a maximum number of idle connections.